A Survey and Benchmark Evaluation for Neural-Network-Based Lossless Universal Compressors Toward Multi-Source Data (Supplementary Materials)

Hui SUN, Huidong MA, Feng LING, Haonan XIE, Yongxia SUN, Liping YI, Meng YAN , Cheng ZHONG, Xiaoguang LIU, Gang WANG

December 18, 2024

This document provides detailed information for benchmark datasets and algorithms, as well as additional experiments. The NNLCB is available at `https://github.com/fahaihi/NNLCB`.

Contents

1 Benchmark Algorithms Details

1.1 NN-Based Compressors

1.1.1 Cmix

Cmix [1] is a neural network based lossless compression algorithm aimed at optimizing compression ratio with the cost of high CPU/memory usage. The NNCP compressor uses thousands of context models followed by an NN-based mixer. We used Cmix V19 to finish the experiments.

```
# compression
cmix -c file file.cmix
# decompression
cmix -d file.cmix file.cmix.out
```

1.1.2 Lstm-compress

Lstm-compress [2] is an LSTM-based lossless compression algorithm that uses the same LSTM module and preprocessing code as Cmix. Lstm-compress currently only supports compression of a single file. In this manuscript, we used Lstm-compress V3. The detailed commands are as follows.

```
# compression
lstm-compress -c file file.lstm
# decompression
lstm-compress -d file.lstm file.lstm.out
```

1.1.3 NNCP

NNCP [3] is a lossless compression algorithm based on LSTM and supports multi-GPU parallel processing. NNCP is an experiment to build a practical lossless data compressor with neural networks. The latest version uses a Transformer model. In this manuscript, we used NNCP V2021-06-01 to finish the experiments. The detailed commands are as follows.

```
# compression
nncp c file file.nncp -T 16 --cuda
# decompression
nncp d file.nncp file.nncp.out -T 16 --cuda
```

1.1.4 DeepZip

DeepZip [4] is a general-purpose (universal) compression algorithm based on recurrent neural networks. It belongs to the static pre-training method. The detailed commands for using DeepZip are shown below.

```
# compression
sh ./compress.sh file file.deepzip bs model
# decompression
sh ./decompress.sh file.deepzip file.deepzip.out bs model
```

1.1.5 DZip

DZip [5] is an upgraded version of DeepZip, with an extra deeper network added to DeepZip to improve compression. DZip includes two compression modes, combined mode and bootstrap mode. The detailed commands of DZip are as follows.

```
# compression
sh ./compress.sh file file.dzip com model
# decompression
sh ./decompress.sh file.dzip file.dzip.out com model
```

1.1.6 TRACE

TRACE [6] is a lossless compression algorithm based on Performer (a Transformer variant.) TRACE uses byte grouping and shared FFNs, and therefore has better execution efficiency. Since the original TRACE puts compression and decompression processes into simultaneous execution, in order to test the performance of compression and decompression separately, we have modified the source files to test the performance of compression or decompression separately.

```
# compression
python compressor.py --source file --comp file.trace
# decompression
python compressor.py --comp file.trace --decomp file.trace.out
```

1.1.7 PAC

PAC [7] is a deep learning based compression algorithm fusing MLP and Ordered Mask. Due to the use of MLP, PAC has a lower computational cost. Again, we separated the compression and decompression process of PAC. The running commands of PAC as shown in the command line below.

```
# compression
python compressor.py --source file --comp file.pac
# decompression
python compressor.py --comp file.pac --decomp file.pac.out
```

1.1.8 LLMZip

LLMZip [8] uses LLaMA as probabilistic predictor in combination with entropy coding (zlib, Token-by-Token and arithmetic coding) to achieve general-purpose lossless compression. The compression and decompression commands of LLMZip are as follows.

```
# compression
torchrun --nproc_per_node 1 LLMzip_run.py --ckpt_dir llama2/llama-2-7b --tokenizer_path
    llama2/tokenizer.model --win_len 511 --text_file file --compression_folder file
    --encode_decode 0
# decompression
torchrun --nproc_per_node 1 LLMzip_run.py --ckpt_dir llama2/llama-2-7b --tokenizer_path
    llama2/tokenizer.model --win_len 511 --text_file file --compression_folder file
    --encode_decode 1
```

1.2 Traditional Compressors

1.2.1 GZip

Gzip [9] is a popular early general-purpose lossless compression program originally written by Jean-loup Gailly for the GNU project. The commands for Gzip are shown below.

```
# compression
gzip -c file > file.gz -9
```

```
# decompression
gzip file.gz -9
```

1.2.2 PBzip2

PBzip2 [10] is a parallel implementation of the Bzip2 block-sorting file compression algorithm that uses pthreads and achieves near-linear speedup on SMP devices. PBzip2 utilizes the Burrows-Wheeler block sorting algorithm for compressing files, along with Huffman coding for efficient text compression. This manuscript uses parallel Bzip2 V1.1.13 to compress data.

```
# compression
pbzip2 -9 -m2000 -p16 -c file > file.bz2
# decompression
pbzip2 -dc -9 -p16 -m2000 file.bz2
```

1.2.3 XZ

XZ Utils [11] is free general-purpose data compression software with a high compression ratio. XZ Utils were written for POSIX-like systems, but also work on some not-so-POSIX systems. XZ Utils are the successor to LZMA Utils. In our experiments, we used XZ V5.5.0. The compression and decompression commands are as follows.

```
# compression
pbzip2 -9 -m2000 -p16 -c file > file.bz2
# decompression
pbzip2 -dc -9 -p16 -m2000 file.bz2
```

1.2.4 BSC

BSC [12] is a high-performance file compressor based on lossless block-ordered data compression algorithm, block-ordered data compression algorithm, high-performance file compressor. This manuscript uses BSC V3.3.2 to compress and decompress data.

```
# compression
bsc e file file.bsc -e2
# decompression
bsc d file.bsc file.bsc.out
```

1.2.5 SnZip

SnZip [13] is a traditional general-purpose lossless compression algorithm based on snappy. It supports a wide range of file formats including framing-format, old framing-format and so on. The default is framing-format. The command line to run SnZip is as follows.

```
# compression
snzip -k -t snzip file
# decompression
snzip -kd -t snzip file.snz
```

1.2.6 Lzma2

Lzma2 [14] improves the multi-threading capability and performance of the LZMA algorithm and better handles incompressible data, so the compression performance is slightly improved. We also used the built-in Lzma2 algorithm in the 7-Zip application.

```
# compression
7zz a -m0=lzma2 -mx9 -mmt16 file.7z file
```

```
# decompression
7zz x -y -mx9 -mmt16 file.7z
```

1.2.7 PPMD

PPMD [15] is a context-based compressor, and its core idea is the Partial Matching Prediction (PPM) algorithm proposed by Cleary and Witten. PPM is a statistical modeling technique that uses a set of previous symbols in the input to predict the next symbol to reduce the output data's entropy. PPM differs from a dictionary because PPM predicts the next symbol instead of trying to find the next symbol in the dictionary to encode. We utilized the PPMD in the 7-Zip to compress data.

```
# compression
7zz a -m0=ppmd -mx9 -mmt16 file.7z file
# decompression
7zz x -y -mx9 -mmt16 file.7z
```

1.2.8 X3

X3 [16, 17] is a dictionary-based compressor, and encodes various code-stream events using an arithmetic coding. The commands of running X3 are as follows.

```
# compression
x3 -zf file file.x3
# decompression
x3 -df file.x3 file.x3.out
```

1.2.9 LZ4-multi

LZ4 [18] is a lossless compression algorithm with compression speeds greater than 500 MB/s per kernel (greater than 0.15 bytes/cycle). Its decoder is extremely fast, up to several GB/s (1 byte/cycle) per kernel. The latest lz4 algorithm supports multi-threaded versions.

```
# compression
lz4 -12 -T16 file file.lz4
# decompression
lz4 -12 -T16 -d file.lz4 file.lz4.reads
```

2 Datasets Acquisition

We evaluated the compression performance of all compressors using 19 real datasets with different sizes and types, those open-sourced datasets from literature [6, 19–31]. The detailed information and download hyperlinks of all benchmark datasets are as follows:

Table 1: The Download Linkages of Experimental Datasets

ID	Dataset	Type	File Size (Byte)	URL
D1	xml [19]	text	5345280	Download xml
D2	ooffice [19]	heterogeneous	6152192	Download ooffice
D3	reymont [19]	text	6627202	Download reymont
D4	sao [19]	homogeneous	7251944	Download sao
D5	x-ray [19]	image	8474240	Download x-ray
D6	mr [19]	image	9970564	Download mr
D7	osdb [19]	heterogeneous	10085684	Download osdb
D8	dickens [19]	text	10192446	Download dickens
D9	samba [19]	heterogeneous	21606400	Download samba
D10	nci [19]	homogeneous	33553445	Download nci
D11	webster [19]	heterogeneous	41458703	Download webster
D12	mozilla [19]	homogeneous	51220480	Download mozilla
D13	enwik8 [20]	text	100000000	Download enwik8
D14	text8 [20]	text	100000000	Download text8
D15	MNIST [24]	image	54880032	Download MNIST
D16	CIFAR-10 [25]	image	186213868	Download CIFAR-10
D17	ImageNet [23]	image	745823247	Download ImageNet
D18	ImageTest [26]	image	470611702	Download ImageTest
D19	Silesia [19]	heterogeneous	211938580	Download Silesia
D20	Backup [6]	heterogeneous	1000000000	Download Backup
D21	enwik9 [20]	text	1000000000	Download enwik9
D22	Book [21]	text	1000000000	Download Book
D23	ESC [22]	audio	220522000	Download ESC
D24	Command [27]	audio	327759206	Download Command
D25	LibriSpeech [28]	audio	359034309	Download LibriSpeech
D26	LJSpeech [29]	audio	293847664	Download LJSpeech
D27	DNACorpus [31]	genome	685597124	Download DNACorpus
D28	ERR7091247 [30]	genome	1926041160	Download ERR7091247

3 Additional Results

We set the metrics *CS*, *CR*, *CT*, *CPM*, *DT*, *DPM* to denote the compressed file size, compression ratio, compression time, compression peak memory overhead, decompression time and decompression peak memory overhead, respectively.

3.1 Detailed Results for Cmix, LLMZip, and X3

3.1.1 Cmix

Table 2: Experimental Results Obtained by Running Cmix on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.233	0.365	114.017	18776.344	102.917	18776.891
D2	1.165	1.589	144.683	19457.617	142.950	19457.723
D3	0.674	0.853	150.567	19389.680	151.417	19389.762
D4	3.554	4.110	167.483	19411.430	172.650	19411.461
D5	3.343	3.308	185.117	18827.934	195.783	18827.828
D6	1.747	1.469	205.983	18661.742	207.283	18661.816
D7	1.871	1.555	235.883	19511.012	233.017	19510.824
D8	1.767	1.453	213.050	18689.773	203.083	18689.563
D9	2.392	0.928	495.650	20330.203	484.983	20335.969
D10	0.757	0.189	545.950	18470.406	536.733	18470.246
D11	4.108	0.831	818.417	19809.230	826.367	19809.277
D12	8.394	1.374	1188.833	21832.168	1219.567	21832.012
D13	14.644	1.228	2026.433	22081.254	2021.233	22081.250
D14	15.597	1.308	1705.917	20580.504	1701.033	20580.387
D15	5.361	0.819	1000.650	20360.113	999.967	20360.168

We executed the compression and decompression process of Cmix on 15 datasets of size less than 100 MBs, and the results are shown in Table 2.

From the results, we can see that the execution time and memory overhead of Cmix on the first 15 datasets are much higher than the other algorithms, which is due to the fact that Cmix mixes more than two thousand models to predict the probability distributions of the target symbols, and of course, Cmix outperformed all other algorithms on compression ratio.

3.1.2 LLMZip

Table 3: Experimental Results Obtained by Running LLMZip on Benchmark Datasets

Datasets	*CS* (Bytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
Cmix						
enwik3	262	2.096	0.084	5605.172	0.084	5605.266
enwik4	2797	2.237	0.268	8389.719	0.267	8389.684
enwik5	23799	1.903	2.067	14949.871	2.083	14949.820
enwik6	193830	1.550	20.451	18565.871	20.201	18566.031
LLMZip						
enwik3	292	2.336	0.951	13700.590	0.535	13700.773
enwik4	668	0.534	3.618	13700.273	3.618	13701.023
enwik5	7653	0.612	28.952	13699.594	28.951	13700.422
enwik6	73855	0.591	313.550	13700.469	312.550	39246.750

LLMZip is a compression algorithm that uses the large language model LLaMa as the probabilistic predictor. Each symbol has to go through the inference operation of the LLM, which also causes it to be much slower than other DL-based and traditional algorithms, and thus cannot complete the benchmark designed in this paper. Therefore, we intercept the first 1,000, 10,000, 100,000, and 1,000,000 bytes of

the English Wikipedia dataset to form the datasets enwik3, enwik4, enwik5, and enwik6 for the test of LLMZip, and select the current algorithm with the optimal compression ratio, Cmix, as the comparison. The experimental results are shown in Table 3.

From the Table 3, we can see that the compression ratio of LLMZip is much better than that of Cmix on the datasets enwik4, enwik5, and enwik6. However, the corresponding computational cost is much higher than that of Cmix, especially the compression/decompression time difference is an order of magnitude.

In conclusion, large language models have great potential for lossless compression, but at the same time the huge computational cost is one of the major challenge.

3.1.3 X3

Table 4: Experimental Results Obtained by Running X3 on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.579	0.909	3.400	22.453	2.133	22.176
D2	2.888	3.938	14.617	105.781	10.950	106.059
D3	1.245	1.576	2.333	27.992	1.083	27.938
D4	5.000	5.784	5.467	100.742	2.767	100.871
D5	4.348	4.304	8.017	59.176	5.050	59.070
D6	2.359	1.984	4.917	53.207	2.850	53.195
D7	3.340	2.778	6.983	51.480	3.583	51.457
D8	2.658	2.188	6.050	42.598	3.350	42.656
D9	5.195	2.017	89.200	163.855	57.417	163.898
D10	1.865	0.466	5.883	53.367	3.233	53.402
D11	8.092	1.637	143.267	177.207	98.583	176.703
D12	18.264	2.991	1315.317	697.516	1135.650	707.898
D13	25.654	2.152	2260.900	575.801	1562.167	575.262
D15	8.946	1.367	189.333	309.422	125.183	310.305
D16	134.085	6.040	2228.700	1237.391	1467.433	1237.711
D23	145.056	5.518	1430.383	1237.141	674.700	1237.035
D25	338.466	7.908	762.233	1134.715	364.483	1134.820
D26	206.139	5.885	889.767	1014.281	487.117	1014.379

X3 is a dictionary-based traditional compression algorithm. Table 4 shows the results of X3 running on only some of the datasets, this is because we tried to execute it multiple times and were unable to complete the test on all datasets on our platform, even with the fastest execution parameters selected. Based on the available results, there is still room for improvement in the compression effectiveness and computational cost of X3.

3.2 Detailed Results for Another Algorithms

3.2.1 NNCP

Table 5: Experimental Results Obtained by Running NNCP on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.449	0.705	34.950	114.145	35.200	113.785
D2	1.959	2.671	40.283	114.004	40.733	113.730
D3	0.865	1.094	43.467	114.137	46.717	113.730
D4	3.645	4.216	50.550	114.090	46.817	113.828
D5	3.401	3.366	54.950	113.926	54.300	113.789
D6	1.877	1.579	63.867	114.066	63.783	113.816
D7	2.080	1.730	64.967	114.129	64.767	113.816
D8	2.280	1.876	65.500	114.063	65.700	113.855
D9	3.837	1.489	137.733	113.988	137.700	113.848
D10	1.142	0.285	214.133	114.082	215.333	113.820
D11	5.677	1.148	291.850	113.910	296.600	113.820
D12	10.600	1.735	361.067	114.039	339.567	113.793
D13	19.909	1.670	564.583	113.977	558.900	113.758
D14	19.734	1.655	885.683	113.988	819.917	113.594
D15	5.625	0.859	376.500	114.035	372.483	113.801
D16	105.120	4.735	1231.917	114.125	1186.133	113.629
D17	600.826	6.757	4210.150	113.992	4197.933	113.848
D18	183.402	3.269	3724.267	113.992	3214.550	113.625
D19	38.790	1.535	1325.033	114.098	1233.300	113.793
D20	474.456	3.980	5675.417	113.965	6593.967	113.785
D21	156.585	1.313	6135.417	113.949	6135.750	113.859
D22	179.739	1.507	6001.867	114.047	5890.100	113.719
D23	95.072	3.616	1410.017	114.051	1459.417	113.828
D24	136.246	3.487	2495.900	114.051	2499.517	113.547
D25	318.479	7.441	2895.517	113.988	2242.367	113.715
D26	148.944	4.251	2419.333	113.902	2258.217	113.512
D27	144.658	1.769	4443.283	113.965	4274.617	113.727
D28	197.778	0.861	11357.467	114.031	11218.567	113.770

3.2.2 Lstm-compress

Table 6: Experimental Results Obtained by Running Lstm-compress on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.455	0.713	14.867	7.199	15.533	7.199
D2	1.964	2.677	22.967	10.145	24.817	10.082
D3	0.956	1.210	24.550	10.082	24.100	10.172
D4	3.769	4.359	26.350	10.168	26.717	10.141
D5	3.407	3.372	31.467	10.070	34.083	10.227
D6	1.876	1.578	36.350	10.191	35.750	10.191
D7	2.111	1.755	40.467	10.137	36.967	10.070
D8	2.334	1.920	28.183	7.180	28.650	7.184
D9	4.090	1.587	80.033	10.141	77.400	10.168
D10	1.460	0.364	88.517	6.590	88.400	6.633
D11	6.100	1.234	114.050	7.230	115.617	7.277
D12	12.416	2.033	183.783	10.168	180.867	10.191
D13	22.097	1.854	344.950	9.238	326.800	9.168
D14	20.936	1.756	237.233	5.867	235.067	6.023
D15	7.170	1.096	198.800	10.176	210.667	10.125
D16	123.055	5.543	708.683	9.969	728.567	10.164
D17	616.534	6.934	2877.867	10.141	2790.967	10.297
D18	405.339	7.225	2007.267	10.066	1867.217	9.977
D19	44.862	1.776	912.033	10.141	773.600	10.199
D20	533.427	4.475	3728.550	10.027	3807.933	10.074
D21	187.254	1.571	3630.217	9.211	3577.133	9.063
D22	203.182	1.704	3583.200	9.113	3551.483	9.008
D23	100.502	3.823	826.600	10.332	789.633	10.363
D24	141.577	3.623	1224.533	9.961	1208.200	9.992
D25	319.552	7.466	1420.183	10.195	1316.067	10.176
D26	156.206	4.459	1362.583	10.008	1118.167	9.988
D27	621.516	7.605	1398.383	5.668	1438.817	5.598
D28	214.544	0.934	4473.067	5.750	4505.500	5.898

3.2.3 DeepZip

Table 7: Experimental Results Obtained by Running DeepZip on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.308	0.484	8.300	1952.535	1.533	1650.543
D2	1.560	2.127	9.717	2011.762	2.200	1675.109
D3	0.639	0.808	10.383	1988.621	2.217	1664.000
D4	3.522	4.074	11.600	2020.703	2.700	1676.785
D5	3.377	3.343	13.500	2053.957	3.083	1698.375
D6	1.692	1.424	15.667	2089.121	3.417	1712.102
D7	1.946	1.619	15.783	2062.762	3.533	1698.344
D8	1.910	1.572	15.867	2069.387	3.050	1693.898
D9	3.457	1.342	34.050	2329.758	7.333	1809.188
D10	1.077	0.269	51.633	2608.441	8.533	1949.051
D11	4.955	1.003	63.550	2798.066	11.833	2009.777
D12	10.092	1.653	80.217	3029.996	17.867	2121.691
D13	19.686	1.651	155.767	4112.680	33.633	2582.793
D14	19.349	1.623	286.967	7167.992	22.983	1701.270
D15	5.086	0.777	149.300	4380.559	17.783	1268.566
D16	442.122	19.917	585.683	12398.508	98.183	2522.895
D17	2044.115	22.991	1413.417	18915.352	412.217	8742.391
D18	1133.470	20.204	815.367	30179.367	216.483	5235.492
D19	43.409	1.718	336.583	6691.273	73.950	3644.801
D20	513.489	4.307	1621.700	24736.355	390.233	11139.488
D21	2145.443	17.997	1738.067	24729.777	494.533	11157.547
D22	196.844	1.651	1605.150	24721.637	343.817	11140.012
D23	89.544	3.406	356.483	6867.934	79.533	3697.969
D24	558.719	14.300	1002.417	21055.316	149.517	3873.039
D25	317.534	7.419	594.250	22975.688	120.517	4171.676
D26	725.921	20.723	522.667	19270.680	138.333	3549.082
D27	152.098	1.861	455.383	17525.047	95.317	8152.754
D28	231.633	1.009	3073.350	120284.188	392.583	19114.172

3.2.4 DeepZip*

Table 8: Experimental Results Obtained by Running DeepZip* on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	2.836	4.450	8.300	1952.535	1.533	1650.543
D2	5.392	7.351	9.717	2011.762	2.200	1675.109
D3	4.470	5.658	10.383	1988.621	2.217	1664.000
D4	7.353	8.506	11.600	2020.703	2.700	1676.785
D5	7.208	7.135	13.500	2053.957	3.083	1698.375
D6	5.524	4.647	15.667	2089.121	3.417	1712.102
D7	5.777	4.805	15.783	2062.762	3.533	1698.344
D8	4.420	3.637	15.867	2069.387	3.050	1693.898
D9	7.289	2.830	34.050	2329.758	7.333	1809.188
D10	3.417	0.854	51.633	2608.441	8.533	1949.051
D11	7.456	1.509	63.550	2798.066	11.833	2009.777
D12	13.923	2.280	80.217	3029.996	17.867	2121.691
D13	23.264	1.952	155.767	4112.680	33.633	2582.793
D14	21.532	1.806	286.967	7167.992	22.983	1701.270
D15	8.917	1.363	149.300	4380.559	17.783	1268.566
D16	445.953	20.089	585.683	12398.508	98.183	2522.895
D17	2047.946	23.034	1413.417	18915.352	412.217	8742.391
D18	1137.302	20.272	815.367	30179.367	216.483	5235.492
D19	47.240	1.870	336.583	6691.273	73.950	3644.801
D20	517.320	4.340	1621.700	24736.355	390.233	11139.488
D21	2149.027	18.027	1738.067	24729.777	494.533	11157.547
D22	200.407	1.681	1605.150	24721.637	343.817	11140.012
D23	93.375	3.552	356.483	6867.934	79.533	3697.969
D24	562.550	14.398	1002.417	21055.316	149.517	3873.039
D25	321.365	7.508	594.250	22975.688	120.517	4171.676
D26	729.752	20.833	522.667	19270.680	138.333	3549.082
D27	152.225	1.863	455.383	17525.047	95.317	8152.754
D28	233.840	1.018	3073.350	120284.188	392.583	19114.172

3.2.5 DZip

Table 9: Experimental Results Obtained by Running DZip on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.284	0.446	14.767	2910.949	5.850	2503.453
D2	1.575	2.148	17.533	2954.473	7.783	2510.242
D3	0.625	0.791	18.567	2965.566	7.767	2523.012
D4	3.519	4.071	21.117	3024.680	9.267	2515.348
D5	3.359	3.326	24.550	3077.281	10.800	2545.047
D6	1.678	1.411	27.767	3193.102	11.800	2537.941
D7	1.931	1.606	28.267	3192.980	12.083	2547.828
D8	1.894	1.559	28.883	3190.648	11.317	2550.816
D9	3.254	1.264	59.733	3912.141	26.333	2657.793
D10	1.054	0.264	90.300	4655.477	34.667	2737.832
D11	4.799	0.971	112.133	5139.852	44.750	2864.516
D12	10.271	1.682	142.533	5725.219	61.150	2971.930
D13	18.774	1.575	210.300	4114.426	84.467	2830.438
D14	18.763	1.574	344.750	7136.996	91.600	1994.465
D15	5.075	0.776	217.800	4368.668	90.583	1562.145
D16	112.007	5.046	847.700	12400.902	328.767	2816.492
D17	672.153	7.560	1709.917	18912.719	852.617	8974.816
D18	197.831	3.526	1012.050	8871.203	475.867	5527.633
D19	39.529	1.565	472.850	6680.633	228.783	3869.578
D20	501.159	4.204	2236.983	24709.680	1171.433	11394.277
D21	179.926	1.509	2108.517	24722.070	963.683	11396.805
D22	186.101	1.561	2152.317	24713.102	1073.750	11390.449
D23	89.188	3.393	634.200	6879.902	294.600	3954.262
D24	144.838	3.707	1109.917	21093.313	391.367	4167.078
D25	317.523	7.419	803.600	22979.719	386.367	4471.723
D26	168.531	4.811	651.350	5667.371	310.700	3847.301
D27	146.872	1.797	787.133	17510.566	515.433	8402.570
D28	206.217	0.898	4081.667	35242.859	1398.883	19259.707

3.2.6 DZip*

Table 10: Experimental Results Obtained by Running DZip* on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	2.812	4.413	14.767	2910.949	5.850	2503.453
D2	5.407	7.372	17.533	2954.473	7.783	2510.242
D3	4.456	5.640	18.567	2965.566	7.767	2523.012
D4	7.350	8.503	21.117	3024.680	9.267	2515.348
D5	7.191	7.118	24.550	3077.281	10.800	2545.047
D6	5.509	4.635	27.767	3193.102	11.800	2537.941
D7	5.762	4.792	28.267	3192.980	12.083	2547.828
D8	4.404	3.624	28.883	3190.648	11.317	2550.816
D9	7.086	2.751	59.733	3912.141	26.333	2657.793
D10	3.394	0.849	90.300	4655.477	34.667	2737.832
D11	7.300	1.477	112.133	5139.852	44.750	2864.516
D12	14.102	2.310	142.533	5725.219	61.150	2971.930
D13	22.353	1.875	210.300	4114.426	84.467	2830.438
D14	20.947	1.757	344.750	7136.996	91.600	1994.465
D15	8.906	1.361	217.800	4368.668	90.583	1562.145
D16	115.838	5.218	847.700	12400.902	328.767	2816.492
D17	675.984	7.603	1709.917	18912.719	852.617	8974.816
D18	201.663	3.595	1012.050	8871.203	475.867	5527.633
D19	43.360	1.716	472.850	6680.633	228.783	3869.578
D20	504.990	4.236	2236.983	24709.680	1171.433	11394.277
D21	183.510	1.539	2108.517	24722.070	963.683	11396.805
D22	189.665	1.591	2152.317	24713.102	1073.750	11390.449
D23	93.020	3.538	634.200	6879.902	294.600	3954.262
D24	148.669	3.805	1109.917	21093.313	391.367	4167.078
D25	321.354	7.508	803.600	22979.719	386.367	4471.723
D26	172.363	4.921	651.350	5667.371	310.700	3847.301
D27	146.999	1.799	787.133	17510.566	515.433	8402.570
D28	208.423	0.908	4081.667	35242.859	1398.883	19259.707

3.2.7 TRACE

Table 11: Experimental Results Obtained by Running TRACE on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.500	0.785	3.233	2356.668	4.733	2386.598
D2	2.222	3.029	4.067	2358.086	5.950	2397.297
D3	0.938	1.187	4.100	2384.641	5.967	2398.012
D4	3.692	4.271	4.900	2378.813	7.233	2399.328
D5	3.467	3.432	5.733	2384.316	8.267	2411.211
D6	2.000	1.682	6.383	2400.691	9.017	2411.973
D7	2.043	1.699	6.283	2371.844	8.900	2431.012
D8	2.432	2.002	6.467	2384.418	9.100	2426.105
D9	4.539	1.762	13.417	2385.129	18.883	2513.098
D10	1.444	0.361	18.683	2373.801	27.717	2596.508
D11	6.573	1.330	24.667	2373.516	35.767	2671.699
D12	12.292	2.013	32.367	2402.551	47.017	2711.332
D13	22.292	1.870	41.217	1981.125	89.783	2522.043
D14	21.239	1.782	40.183	1967.715	73.167	2554.129
D15	6.251	0.955	33.733	1350.297	52.183	1824.648
D16	114.929	5.177	124.533	3365.840	191.350	3950.434
D17	640.493	7.204	344.767	12451.441	713.317	12995.129
D18	196.117	3.496	201.000	7970.629	363.017	8564.500
D19	44.749	1.771	83.317	3796.121	172.833	4336.215
D20	542.016	4.547	455.667	16573.621	1108.517	16829.383
D21	185.476	1.556	634.783	16570.969	1039.200	17112.047
D22	212.338	1.781	422.150	16572.598	889.383	17118.887
D23	103.291	3.929	93.933	3935.559	197.567	4476.988
D24	143.144	3.664	143.683	5661.016	270.150	6248.000
D25	319.317	7.461	170.767	6167.133	289.333	6746.480
D26	157.156	4.486	170.133	5104.770	238.450	5696.980
D27	152.830	1.870	290.083	11475.219	572.033	12016.211
D28	226.757	0.988	767.400	31565.016	1417.767	32155.965

3.2.8 PAC

Table 12: Experimental Results Obtained by Running PAC on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.522	0.819	3.983	2387.141	5.417	2397.559
D2	2.022	2.757	5.000	2374.750	6.800	2401.742
D3	0.960	1.215	4.817	2407.387	6.950	2408.152
D4	3.686	4.264	5.983	2383.949	7.900	2426.754
D5	3.492	3.456	7.067	2380.254	8.917	2430.973
D6	2.129	1.791	7.750	2381.500	9.833	2431.480
D7	2.146	1.785	7.033	2389.156	9.883	2435.777
D8	2.363	1.945	6.967	2387.855	10.083	2452.203
D9	3.969	1.541	14.467	2395.480	21.167	2537.496
D10	1.387	0.347	21.517	2379.309	31.233	2593.410
D11	5.908	1.195	28.533	2362.906	39.767	2659.699
D12	11.156	1.827	36.617	2395.727	50.483	2732.488
D13	20.210	1.695	51.567	1970.391	89.450	2495.047
D14	19.379	1.626	50.717	1970.352	85.633	2495.000
D15	7.239	1.106	26.033	1275.289	44.300	1747.582
D16	99.209	4.469	103.117	3366.324	167.433	3887.234
D17	618.911	6.961	441.067	12441.035	671.167	12958.430
D18	188.614	3.362	224.750	7970.477	373.550	8493.777
D19	40.525	1.604	108.917	3785.918	190.683	4301.777
D20	500.732	4.200	546.600	16561.676	863.167	17079.383
D21	164.170	1.377	444.583	16561.211	782.433	17084.098
D22	189.540	1.590	491.050	16562.176	787.517	17075.449
D23	110.430	4.201	112.150	3924.051	0.050	372.973
D24	151.165	3.869	150.333	5650.383	246.450	6183.984
D25	318.355	7.438	195.567	6161.402	306.167	6699.219
D26	167.597	4.784	196.117	5104.531	247.233	5623.844
D27	147.308	1.802	300.583	11464.469	500.250	12009.027
D28	194.969	0.849	870.967	31566.375	1448.133	32079.230

3.2.9 GZip

Table 13: Experimental Results Obtained by Running GZip on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.632	0.991	0.017	1.953	0.017	1.688
D2	2.947	4.019	0.017	1.875	0.017	1.863
D3	1.736	2.198	0.033	1.914	0.017	1.840
D4	5.080	5.877	0.017	1.941	0.017	1.656
D5	5.758	5.700	0.017	1.945	0.017	1.699
D6	3.504	2.948	0.033	1.859	0.017	1.844
D7	3.544	2.948	0.017	1.773	0.017	1.840
D8	3.673	3.023	0.033	2.031	0.017	1.770
D9	5.158	2.002	0.033	1.871	0.017	1.762
D10	2.849	0.712	0.050	1.855	0.017	1.691
D11	11.503	2.327	0.067	1.754	0.017	1.898
D12	18.114	2.967	0.167	1.828	0.017	1.664
D13	34.757	2.916	0.133	1.863	0.017	1.660
D14	31.517	2.644	0.167	1.758	0.017	1.656
D15	10.811	1.652	0.333	1.895	0.017	1.867
D16	162.547	7.322	0.167	1.852	0.033	1.773
D17	702.067	7.896	0.550	1.945	0.117	1.844
D18	330.164	5.885	0.500	1.809	0.100	1.867
D19	64.517	2.554	0.400	1.957	0.033	1.875
D20	743.469	6.237	0.983	1.965	0.183	1.730
D21	307.648	2.581	1.050	1.805	0.150	1.871
D22	342.778	2.875	1.683	1.863	0.150	1.902
D23	156.003	5.934	0.217	1.746	0.050	1.773
D24	225.450	5.770	0.483	1.883	0.067	1.773
D25	321.477	7.511	0.233	1.824	0.050	1.781
D26	240.003	6.851	0.283	1.895	0.067	1.898
D27	177.455	2.171	9.900	1.816	0.100	1.902
D28	370.926	1.616	9.483	1.871	0.233	1.777

3.2.10 PBzip2

Table 14: Experimental Results Obtained by Running PBzip2 on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.418	0.656	0.017	45.313	0.017	6.902
D2	2.726	3.716	0.017	51.441	0.017	29.777
D3	1.188	1.504	0.017	55.328	0.017	33.676
D4	4.713	5.452	0.017	60.984	0.017	40.992
D5	3.864	3.824	0.017	69.965	0.017	43.566
D6	2.343	1.971	0.017	64.012	0.017	39.199
D7	2.674	2.224	0.017	82.059	0.017	51.309
D8	2.669	2.196	0.017	83.395	0.017	51.625
D9	4.363	1.693	0.017	117.348	0.017	63.711
D10	1.766	0.441	0.017	130.473	0.017	78.910
D11	8.248	1.668	0.017	132.527	0.017	87.598
D12	17.066	2.794	0.017	132.715	0.017	89.582
D13	27.671	2.321	0.017	138.285	0.017	95.387
D14	25.178	2.112	0.033	137.379	0.017	106.250
D15	9.580	1.464	0.017	68.574	0.017	55.383
D16	140.276	6.319	0.033	152.461	0.033	115.441
D17	696.059	7.828	0.117	161.840	0.083	120.848
D18	260.493	4.643	0.083	152.891	0.050	138.977
D19	52.120	2.063	0.050	142.414	0.017	120.184
D20	712.635	5.978	0.133	167.500	0.083	142.363
D21	242.298	2.032	0.117	141.875	0.067	106.531
D22	255.441	2.142	0.117	141.996	0.067	105.223
D23	133.081	5.062	0.050	150.148	0.033	134.223
D24	185.766	4.754	0.067	148.301	0.033	118.242
D25	322.556	7.536	0.083	156.027	0.050	120.512
D26	205.676	5.872	0.067	151.281	0.033	120.789
D27	171.842	2.102	0.083	141.438	0.067	96.395
D28	303.885	1.324	0.200	123.426	0.083	98.289

3.2.11 XZ

Table 15: Experimental Results Obtained by Running XZ on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.415	0.650	0.050	114.277	0.017	12.848
D2	2.315	3.156	0.050	128.152	0.017	16.129
D3	1.255	1.588	0.067	129.168	0.017	16.117
D4	4.221	4.882	0.067	140.465	0.017	20.324
D5	4.283	4.239	0.050	152.355	0.017	22.586
D6	2.624	2.208	0.117	157.828	0.017	23.574
D7	2.713	2.256	0.083	166.020	0.017	24.145
D8	2.700	2.222	0.100	154.422	0.017	24.340
D9	3.566	1.384	0.183	276.871	0.017	46.797
D10	1.382	0.345	0.367	342.055	0.017	67.684
D11	7.981	1.614	0.467	466.852	0.017	89.082
D12	12.757	2.089	0.400	568.453	0.033	112.770
D13	23.681	1.986	1.300	793.680	0.033	185.293
D14	22.145	1.858	1.850	748.543	0.033	183.594
D15	8.658	1.323	1.083	599.273	0.017	115.633
D16	133.348	6.007	2.417	986.711	0.233	377.348
D17	695.042	7.817	2.183	4100.766	0.283	1511.156
D18	268.436	4.785	3.067	2707.406	0.200	874.066
D19	46.174	1.828	2.500	1079.629	0.083	320.328
D20	605.515	5.079	2.533	4927.270	0.250	1683.922
D21	204.234	1.713	3.933	4525.984	0.083	1412.191
D22	242.129	2.031	4.867	4563.594	0.067	1456.449
D23	135.540	5.155	1.883	1250.883	0.150	418.043
D24	184.560	4.724	3.100	1845.930	0.200	558.055
D25	322.676	7.539	2.000	2013.664	0.150	753.809
D26	205.467	5.866	2.533	1826.195	0.233	551.094
D27	147.148	1.800	6.050	3221.172	0.067	994.086
D28	266.885	1.162	9.467	8231.441	0.100	2581.180

3.2.12 BSC

Table 16: Experimental Results Obtained by Running BSC on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.363	0.570	0.017	28.840	0.017	30.887
D2	2.435	3.319	0.033	37.074	0.033	40.234
D3	0.935	1.183	0.017	39.902	0.017	43.121
D4	4.459	5.158	0.033	43.094	0.033	46.367
D5	3.579	3.542	0.033	48.883	0.033	52.141
D6	2.100	1.767	0.033	45.715	0.017	46.375
D7	2.107	1.752	0.033	56.637	0.017	59.883
D8	2.150	1.769	0.033	56.703	0.017	59.898
D9	3.710	1.440	0.033	98.781	0.033	98.930
D10	1.191	0.297	0.033	121.480	0.017	121.688
D11	6.337	1.282	0.067	133.305	0.033	136.543
D12	15.219	2.492	0.083	152.141	0.083	131.668
D13	21.257	1.783	0.150	164.598	0.117	157.680
D14	19.868	1.667	0.217	165.719	0.150	158.387
D15	7.959	1.217	0.067	127.477	0.067	111.199
D16	127.966	5.765	0.783	164.313	0.817	157.621
D17	672.743	7.566	2.583	163.863	2.617	159.633
D18	228.710	4.077	1.567	167.207	1.450	158.430
D19	45.155	1.787	0.400	165.656	0.317	157.305
D20	642.982	5.393	2.983	165.586	2.750	161.633
D21	184.895	1.551	1.467	164.766	1.017	157.652
D22	208.133	1.745	1.567	166.797	1.117	158.359
D23	125.951	4.791	0.583	160.953	0.533	152.785
D24	167.556	4.288	1.150	165.645	1.017	158.215
D25	320.907	7.498	1.700	159.426	1.667	156.297
D26	188.324	5.376	1.183	165.559	1.067	153.766
D27	150.064	1.836	1.150	164.438	0.817	158.465
D28	247.923	1.080	3.167	167.516	2.117	140.301

3.2.13 SnZip

Table 17: Experimental Results Obtained by Running SnZip on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	1.240	1.945	0.017	3.141	0.017	2.949
D2	4.092	5.580	0.017	3.230	0.017	3.152
D3	3.057	3.869	0.017	3.156	0.017	3.152
D4	6.140	7.102	0.017	3.078	0.017	3.266
D5	8.067	7.985	0.017	3.223	0.017	2.980
D6	5.150	4.333	0.017	3.316	0.017	2.941
D7	5.079	4.224	0.017	3.086	0.017	3.133
D8	5.960	4.906	0.017	3.152	0.017	2.965
D9	7.645	2.968	0.017	3.219	0.017	3.109
D10	5.841	1.460	0.017	3.285	0.017	3.063
D11	19.062	3.857	0.017	3.309	0.017	3.117
D12	25.342	4.150	0.017	3.113	0.017	2.984
D13	53.351	4.475	0.033	3.086	0.033	3.133
D14	51.297	4.303	0.033	3.102	0.033	2.961
D15	15.626	2.388	0.017	3.070	0.017	3.164
D16	176.024	7.930	0.017	3.172	0.017	2.984
D17	711.190	7.999	0.017	3.172	0.017	3.105
D18	410.976	7.326	0.050	3.113	0.033	3.109
D19	96.736	3.829	0.050	3.223	0.033	3.164
D20	846.365	7.100	0.100	3.277	0.050	2.980
D21	479.171	4.020	0.300	3.129	0.150	3.211
D22	557.717	4.678	0.333	3.203	0.183	3.090
D23	185.278	7.048	0.017	3.113	0.017	3.211
D24	288.635	7.387	0.033	3.230	0.017	3.203
D25	325.677	7.609	0.017	3.113	0.017	3.109
D26	279.587	7.982	0.017	3.230	0.017	3.043
D27	300.928	3.682	0.217	3.195	0.133	3.078
D28	614.493	2.676	0.417	3.293	0.267	3.121

3.2.14 Lzma2

Table 18: Experimental Results Obtained by Running Lzma2 on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.434	0.680	0.033	72.598	0.017	10.652
D2	2.313	3.154	0.033	79.758	0.017	13.430
D3	1.257	1.591	0.050	83.848	0.017	12.637
D4	4.209	4.869	0.033	89.227	0.017	16.246
D5	4.272	4.229	0.033	115.832	0.017	17.355
D6	2.621	2.205	0.050	128.418	0.017	17.141
D7	2.720	2.262	0.050	129.332	0.017	17.586
D8	2.700	2.222	0.067	130.445	0.017	17.613
D9	3.586	1.392	0.083	260.414	0.017	29.402
D10	1.661	0.415	0.133	362.805	0.017	38.707
D11	8.000	1.618	0.267	430.871	0.017	52.746
D12	12.726	2.084	0.200	514.559	0.017	66.824
D13	23.711	1.988	0.850	682.023	0.017	124.434
D14	22.149	1.858	1.583	682.176	0.033	123.070
D15	8.785	1.343	0.333	545.938	0.017	66.285
D16	133.627	6.020	1.150	685.051	0.117	318.336
D17	694.413	7.810	0.667	3154.934	0.167	1422.199
D18	268.524	4.786	1.950	1884.727	0.183	727.109
D19	46.455	1.839	1.217	684.805	0.050	254.590
D20	606.015	5.083	1.133	3890.695	0.167	1575.250
D21	204.840	1.718	2.867	3489.426	0.067	1167.371
D22	241.677	2.027	3.717	3525.965	0.067	1205.047
D23	135.099	5.139	0.833	685.141	0.117	352.543
D24	184.310	4.717	2.650	1605.215	0.150	505.125
D25	322.307	7.530	0.950	1832.605	0.083	675.258
D26	204.908	5.850	1.850	1334.871	0.183	493.852
D27	147.151	1.800	6.700	2550.410	0.067	809.141
D28	275.081	1.198	5.533	6614.945	0.117	2123.047

3.2.15 PPMD

Table 19: Experimental Results Obtained by Running PPMD on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.343	0.538	0.017	134.531	0.017	133.602
D2	2.381	3.245	0.033	134.969	0.033	134.219
D3	1.005	1.272	0.017	134.969	0.033	134.141
D4	4.562	5.277	0.050	102.859	0.050	101.992
D5	3.673	3.635	0.050	123.859	0.050	122.844
D6	2.213	1.862	0.033	140.684	0.033	140.055
D7	2.274	1.891	0.033	202.895	0.033	201.977
D8	2.175	1.789	0.050	246.914	0.050	245.766
D9	3.639	1.412	0.050	263.074	0.067	262.262
D10	1.416	0.353	0.033	263.176	0.050	262.293
D11	6.621	1.339	0.117	262.930	0.133	262.141
D12	15.175	2.485	0.150	263.105	0.167	262.336
D13	22.099	1.853	0.350	263.180	0.367	262.348
D14	20.420	1.713	0.467	263.395	0.483	262.375
D15	8.563	1.309	0.200	263.203	0.200	262.367
D16	131.200	5.910	1.700	263.199	1.833	262.371
D17	682.264	7.673	8.083	263.410	8.167	262.410
D18	248.956	4.438	3.333	263.426	3.600	262.195
D19	45.882	1.816	0.750	263.434	0.850	262.297
D20	680.304	5.706	6.867	263.168	6.150	262.273
D21	191.069	1.602	3.633	263.297	4.100	262.418
D22	214.733	1.801	5.150	263.109	6.483	262.383
D23	130.039	4.946	1.400	263.156	1.450	262.223
D24	174.001	4.453	2.483	263.211	2.850	262.227
D25	327.188	7.645	4.450	263.441	4.933	262.246
D26	195.713	5.587	2.833	263.383	3.133	262.359
D27	158.915	1.944	3.817	262.926	4.133	262.406
D28	277.780	1.210	7.433	263.387	7.733	262.246

3.2.16 LZ4-multi

Table 20: Experimental Results Obtained by Running LZ4-multi on Benchmark Datasets

Datasets	*CS* (MegaBytes)	*CR* (bits/base)	*CT* (Minutes)	*CPM* (MegaBytes)	*DT* (Minutes)	*DPM* (MegaBytes)
D1	0.726	1.140	0.017	8.109	0.017	7.445
D2	3.374	4.600	0.017	10.984	0.017	10.680
D3	1.971	2.494	0.017	10.602	0.017	9.973
D4	5.408	6.256	0.017	13.508	0.017	16.223
D5	6.848	6.778	0.017	16.301	0.017	20.172
D6	3.998	3.364	0.033	14.094	0.017	15.234
D7	3.774	3.139	0.017	13.820	0.017	15.078
D8	4.182	3.442	0.017	14.332	0.017	14.613
D9	5.825	2.262	0.033	27.215	0.017	20.438
D10	3.460	0.865	0.017	38.273	0.017	17.145
D11	13.209	2.673	0.017	52.863	0.017	28.012
D12	21.015	3.442	0.050	66.531	0.017	31.414
D13	40.043	3.359	0.033	97.676	0.017	31.473
D14	37.157	3.117	0.033	95.270	0.017	31.055
D15	11.989	1.832	0.200	68.129	0.017	26.840
D16	168.490	7.590	0.033	173.766	0.017	33.363
D17	701.494	7.890	0.067	257.613	0.017	33.832
D18	373.514	6.658	0.117	460.441	0.017	33.605
D19	73.797	2.921	0.067	115.992	0.017	33.125
D20	793.085	6.653	0.117	280.258	0.033	37.563
D21	355.190	2.980	0.167	182.148	0.033	31.461
D22	388.988	3.263	0.217	173.086	0.033	31.516
D23	174.370	6.633	0.033	183.230	0.017	33.410
D24	254.529	6.514	0.067	212.379	0.017	33.820
D25	322.461	7.534	0.033	211.270	0.017	33.488
D26	270.723	7.728	0.033	209.367	0.017	33.434
D27	223.475	2.734	1.133	152.094	0.033	31.066
D28	453.821	1.977	1.200	167.094	0.050	31.004

References

[1] Byron Knoll. Cmix. 2014, `https://github.com/byronknoll/cmix`.

[2] Byron Knoll. Lstm-compress. 2016, `https://github.com/byronknoll/lstm-compress`.

[3] Bellard Fabrice. NNCP. 2019, `https://bellard.org/nncp/nncp.pdf`.

[4] Goyal M, Tatwawadi K, Chandak S, et al. Deepzip: Lossless data compression using recurrent neural networks. arXiv preprint arXiv:1811.08162, 2018.

[5] Goyal M, Tatwawadi K, Chandak S, et al. DZip: Improved general-purpose lossless compression based on novel neural network modeling. In: Proceedings of 2021 Data Compression Conference (DCC). 2021: 153-162.

[6] Mao Y, Cui Y, Kuo T W, et al. TRACE: A Fast Transformer-based General-Purpose Lossless Compressor. In: Proceedings of ACM Web Conference 2022(WWW). 2022: 1829-1838.

[7] Mao Y, Li J, Cui Y, et al. Faster and Stronger Lossless Compression with Optimized Autoregressive Framework. In: Proceedings of the 2023 60th ACM/IEEE Design Automation Conference (DAC). 2023: 1-6.

[8] Valmeekam C S K, Narayanan K, Kalathil D, et al. LLMZip: Lossless Text Compression using Large Language Models. 2024, `https://github.com/vcskaushik/LLMzip`.

[9] Jean-loup Gailly, Mark Adler. Gzip. 1992, https://www.gnu.org/software/gzip.

[10] PBzip2 official website. PBzip2. 2009, https://launchpad.net/pbzip2.

[11] Lasse Collin. The official website of the XZ compressor. 2015, https://tukaani.org/xz/.

[12] IlyaGrebnov. High performance block-sorting data compression library. 2011, https://github.com/IlyaGrebnov/libbsc.

[13] SnZip official website. 2016. https://github.com/kubo/snzip.

[14] Lzma2 official website. Lzma2. 2009, https://www.7-zip.org.

[15] PPMD official website. PPMD. 2010, https://www.7-zip.org.

[16] Barina D, Klima O. x3: Lossless data compressor. In: Proceedings of the 2022 Data Compression Conference, 2022, p. 441.

[17] Barina D. Experimental lossless data compressor. Microprocessors and Microsystems, 2023, 98: 104803.

[18] LZ4-multi. LZ4 official website. https://github.com/lz4/lz4.

[19] Sebastian Deorowicz. Silesia Corpus. 1985, https://sun.aei.polsl.pl//~sdeor/index.php?page=silesia.

[20] Matt Mahoney. Large Text Compression Benchmark. 2006, http://www.mattmahoney.net/dc/text.html.

[21] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. Advances in Neural Information Processing Systems, 2017, 30.

[22] Piczak K J. ESC: Dataset for environmental sound classification. In: Proceedings of the 23rd ACM international conference on Multimedia(MM). 2015: 1015-1018.

[23] Russakovsky O, Deng J, Su H, et al. Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 2015, 115: 211-252.

[24] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[25] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/cifar.html

[26] Image Compression Benchmark official website. http://imagecompression.info/test_images/

[27] Warden P. Speech commands: A dataset for limited-vocabulary speech recognition. arXiv preprint arXiv:1804.03209, 2018.

[28] Panayotov V, Chen G, Povey D, et al. Librispeech: an asr corpus based on public domain audio books. 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2015: 5206-5210.

[29] Ito K, Johnson L. The LJ Speech Dataset. 2017. https://keithito.com/LJ-Speech-Dataset

[30] Geer L Y, Marchler-Bauer A, Geer R C, et al. The NCBI biosystems database. Nucleic acids research, 2010, 38(suppl_1): D492-D496.

[31] Pratas D, Pinho A J. A DNA sequence corpus for compression benchmark. In: Proceedings of the 12th International Conference of Practical Applications of Computational Biology and Bioinformatics. 2019: 208-215.